Automatic Pattern Acquisition for Japanese Information Extraction

Kiyoshi Sudo
Computer Science Department
New York University
715 Broadway, 7th floor,
New York, NY 10003 USA
sudo@cs.nyu.edu

Satoshi Sekine
Computer Science Department
New York University
715 Broadway, 7th floor,
New York, NY 10003 USA
sekine@cs.nyu.edu

Ralph Grishman
Computer Science Department
New York University
715 Broadway, 7th floor,
New York, NY 10003 USA
grishman@cs.nyu.edu

ABSTRACT

One of the central issues for information extraction is the cost of customization from one scenario to another. Research on the automated acquisition of patterns is important for portability and scalability. In this paper, we introduce Tree-Based Pattern representation where a pattern is denoted as a path in the dependency tree of a sentence. We outline the procedure to acquire Tree-Based Patterns in Japanese from un-annotated text. The system extracts the relevant sentences from the training data based on TF/IDF scoring and the common paths in the parse tree of relevant sentences are taken as extracted patterns.

Keywords

Information Extraction, Pattern Acquisition

1. INTRODUCTION

Information Extraction (IE) systems today are commonly based on pattern matching. New patterns need to be written when we customize an IE system for a new scenario (extraction task); this is costly if done by hand. This has led to recent research on automated acquisition of patterns from text with minimal pre-annotation. Riloff [4] reported a successful result for her procedure that needs only a pre-classified corpus. Yangarber [6] developed a procedure for unannotated natural language texts.

One of their common assumption is that the relevant documents include good patterns. Riloff implemented this idea by applying the pre-defined heuristic rules to pre-classified (relevant) documents and Yangarber advanced further so that the system can classify the documents by itself given *seed patterns* specific to a scenario and then find the best patterns from the relevant document set.

Considering how they represent the patterns, we can see that, in general, Riloff and Yangarber relied on the sentence structure of English. Riloff's predefined heuristic rules are based on syntactic structures, such as "<subj> active-verb" and "active-verb <dobj>". Yangarber used triples of a predicate and some of its arguments, such as "<pred> <subj> <obj>".

The Challenges

Our careful examination of Japanese revealed some of the challenges for automated acquisition of patterns and information extraction on Japanese(-like) language and other challenges which arise regardless of the languages.

Free Word-ordering

Free word order is one of the most significant problems in analyzing Japanese. To capture all the possible patterns given a predicate and its arguments, we need to permute the arguments and list all the patterns separately. For example, for "<subj> <dobj> <iobj> <predicate>" with the constraint that the predicate comes last in the sentence, there would be six possible patterns (permutations of three arguments). The number of patterns to cover even simple facts would rise unacceptably high.

Flexible case marking system

There is also a difficulty in a language with a flexible case marking system, like Japanese. In particular, we found that, in Japanese, some of the arguments that are usually marked as object in English were variously marked by different post-positions, and some case markers (postpositions) are used for marking more than one grammatical category in different situations. For example, the topic marker in Japanese, "wa", can mark almost any entity that would have been variously marked in English. It is difficult to deal with this variety by simply fixing the number of arguments of a predicate for creating patterns in Japanese.

Relationships beyond direct predicate-argument

Furthermore, we may want to capture the relationship between a predicate and a modifier of one of its arguments. In previous approaches, one had to introduce an ad hoc frame for such a relationship, such as "verb obj [PP <head-noun>]", to extract the relationship between "to assume" and "<organization>" in the sentence "<person> will assume the <post> of <organization>".

Relationships beyond clausal boundaries

Another problem lies in relationships beyond clause boundaries, especially if the event is described in a subordinate clause. For example, for a sentence like "<organization> announced that <person> retired from <post>," it is hard to find a relationship between <organization> and the event of retiring without the global view

.

Report Documentation Page		*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **2001**	2. REPORT TYPE	3. DATES COVERED **00-00-2001 to 00-00-2001**
4. TITLE AND SUBTITLE **Automatic Pattern Acquisition for Japanese Information Extraction**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Computer Science ,New York University,715 Broadway,New York,NY,10003**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES **7**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

from the predicate “announce”.

These problems lead IE systems to fail to capture some of the arguments needed for filling the template. Overcoming the problems above makes the system capable of finding more patterns from the training data, and therefore, more slot-fillers in the template.

In this paper, we introduce Tree-based pattern representation and consider how it can be acquired automatically.

2. TREE-BASED PATTERN REPRESENTATION (TBP)

Definition

Tree-based representation of patterns (TBP) is a representation of patterns based on the dependency tree of a sentence. A pattern is defined as a path in the dependency tree passing through zero or more intermediate nodes within the tree. The dependency tree is a directed tree whose nodes are *bunsetsu*s or phrasal units, and whose directed arcs denote the dependency between two *bunsetsu*s: A→B denotes A’s dependency on B (e.g. A is a subject and B is a predicate.) Here dependency relationships are not limited to just those between a case-marked element and a predicate, but also include those between a modifier and its head element, which covers most relationships within sentences. [1]

TBP for Information Extraction

Figure 2 shows how TBP is used in comparison with the word-order based pattern, where A...F in the left part of the figure is a sequence of the phrasal units in a sentence appearing in this order and the tree in the right part is its dependency tree. To find the relationship between B→F, a word-order based pattern needs a dummy expression to hold C, D and E, while TBF can denote the direct relationship as B→F. TBP can also represent a complicated pattern for a node which is far from the root node in the dependency tree, like C→D→E, which is hard to represent without the sentence structure.

For matching with TBP, the target sentence should be parsed into a dependency tree. Then all the predicates are detected and the subtrees which have a predicate node as a root are traversed to find a match with a pattern.

Benefit of TBP

TBP has some advantages for pattern matching over the surface word-order based patterns in addressing the problems mentioned in the previous section:

- Free word-order problem

  TBP can offer a direct representation of the dependency relationship even if the word-order is different.

- Free case-marking problem

  TBP can freely traverse the whole dependency tree and find any significant path as a pattern. It does not depend on predefined case-patterns as Riloff [4] and Yangarber [6] did.

- Indirect relationships

  TBP can find indirect relationships, such as the relationship between a predicate and the modifier of the argument of the predicate. For example, the pattern
  “<organization>$\xrightarrow{\text{of}}$<post>$\xrightarrow{\text{to}}$appoint” can capture the relationship between “<organization>” and “to be appointed” in the sentence
  “<person> was appointed to <post> of <organization>.”

- Relationships beyond clausal boundaries

  TBP can capture relationships beyond clausal boundaries. The pattern “<post>$\xrightarrow{\text{to}}$appoint$\xrightarrow{\text{COMP}}$announce” can find the relationship between “<post>” and “to announce”. This relationship, later on, can be combined with the relationship “<organization>” and “to announce” and merged into one event.

3. ALGORITHM

In this section, we outline our procedure for automatic acquisition of patterns. We employ a cascading procedure, as is shown in Figure 3. First, the original documents are processed by a morphological analyzer and NE-tagger. Then the system retrieves the relevant documents for the scenario as a *relevant document set*. The system, further, selects a set of relevant sentences as a *relevant sentence set* from those in the *relevant document set*. Finally, all the sentences in the *relevant sentence set* are parsed and the paths in the dependency tree are taken as patterns.

3.1 Document Preprocessing

Morphological analysis and Named Entity (NE) tagging is performed on the training data at this stage. We used JUMAN [2] for the former and a NE-system which is based on a decision tree algorithm [5] for the latter. Also the part-of-speech information given by JUMAN is used in the later stages.

3.2 Document Retrieval

The system first retrieves the documents that describe the events of the scenario of interest, called the *relevant document set*. A set of narrative sentences describing the scenario is selected to create a query for the retrieval. For this experiment, we set the size of the *relevant document set* to 300 and retrieved the documents using CRL’s stochastic-model-based IR system [3], which performed well in the IR task in IREX, Information Retrieval and Extraction evaluation project in Japan [2]. All the sentences used to create the patterns are retrieved from this *relevant document set*.

3.3 Sentence Retrieval

The system then calculates the TF/IDF-based score of relevance to the scenario for each sentence in the *relevant document set* and retrieves the *n* most relevant sentences as the source of the patterns, where *n* is set to 300 for this experiment. The retrieved sentences will be the source for pattern extraction in the next subsection.

First, the TF/IDF-based score for every word in the *relevant document set* is calculated. TF/IDF score of word *w* is:

$$score(w) = \begin{cases} TF(w) \cdot \frac{\frac{log(N+0.5)}{DF(w)}}{log(N+1)} & \text{if } w \text{ is Noun, Verb or Named Entity} \\ 0 & \text{otherwise} \end{cases}$$

where *N* is the number of documents in the collection, *TF*(*w*) is the term frequency of *w* in the relevant document set and *DF*(*w*) is the document frequency of *w* in the collection.

Second, the system calculates the score of each sentence based on the score of its words. However, unusually short sentences and

[1] In this paper, we used the Japanese parser KNP [1] to obtain the dependency tree of a sentence.

[2] IREX Homepage: http://cs.nyu.edu/cs/projects/proteus/irex

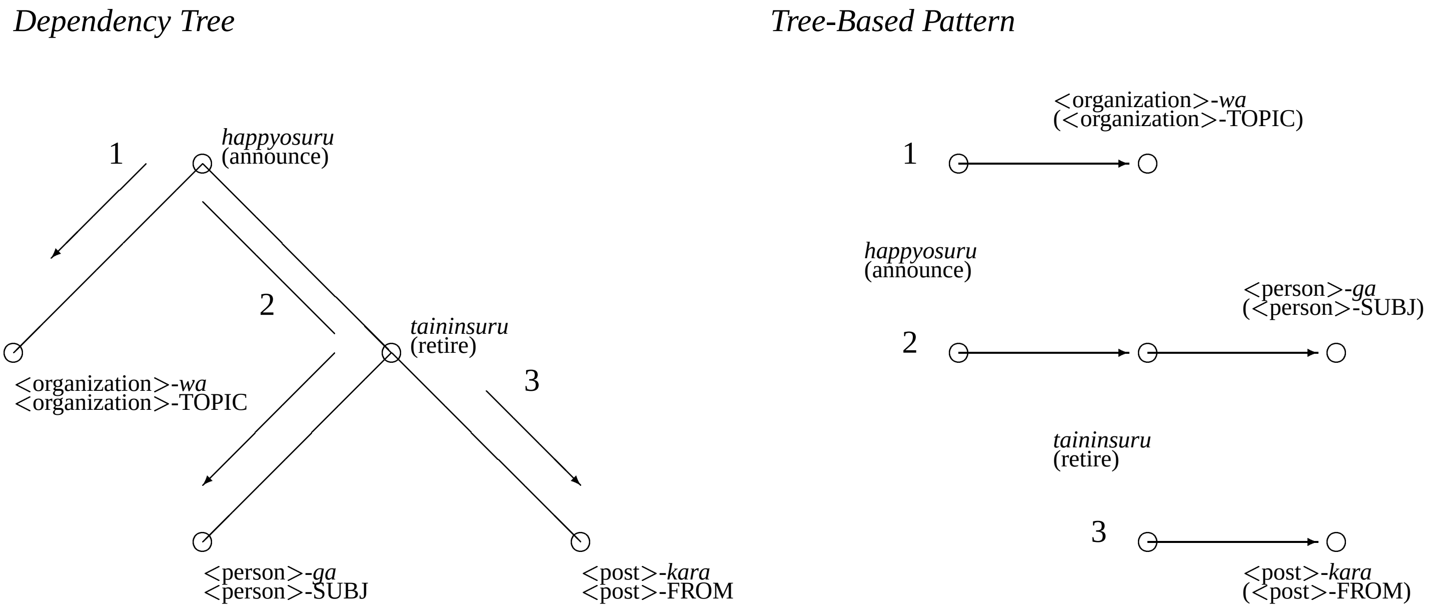


Figure 1: Tree-Based Pattern Representation

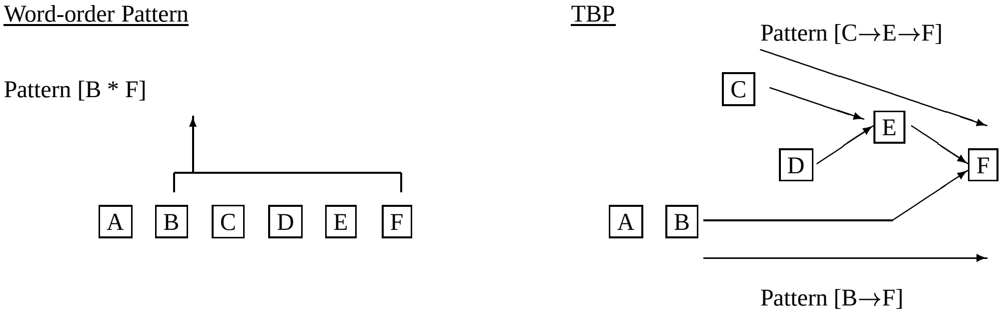


Figure 2: Extraction using Tree-Based Pattern Representation

unusually long sentences will be penalized. The TF/IDF score of sentence *s* is:

$$score(s) = \frac{\sum_{w \in s} score(w)}{length(s) + |length(s) - AVE|}$$

where *length*(*s*) is the number of words in *s*, and *AVE* is the average number of words in a sentence.

3.4 Pattern Extraction

Based on the dependency tree of the sentences, patterns are extracted from the relevant sentences retrieved in the previous subsection. Figure 4 shows the procedure. First, the retrieved sentence is parsed into a dependency tree by KNP [1] (Stage 1). This stage also finds the predicates in the tree. Second, the system takes all the predicates in the tree as the roots of their own subtrees, as is shown in (Stage 2). Then each path from the root to a node is extracted, and these paths are collected and counted across all the relevant sentences. Finally, the system takes those paths with frequency higher than some threshold as extracted patterns. Figure 5 shows examples of the acquired patterns.

4. EXPERIMENT

It is not a simple task to evaluate how good the acquired patterns are without incorporating them into a complete extraction system with appropriate template generation, etc. However, finding a match of the patterns and a portion of the test sentences can be a good measure of the performance of patterns.

The task for this experiment is to find a *bunsetsu*, a phrasal unit, that includes slot-fillers by matching the pattern to the test sentence. The performance is measured by recall and precision in terms of the number of slot-fillers that the matched patterns can find; these are calculated as follows.

$$Recall = \frac{\#\ of\ Matched\ Relevant\ SlotFillers}{\#\ of\ All\ Relevant\ SlotFillers}$$

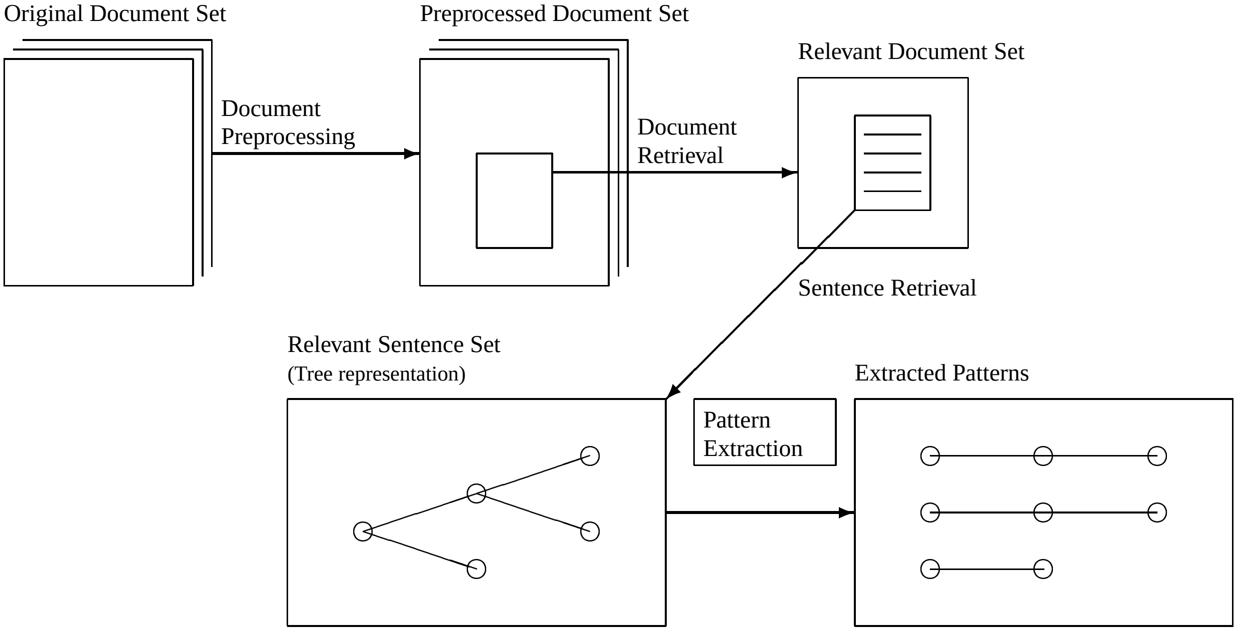


Figure 3: Pattern Acquisition Procedure Overall Process

$$Precision = \frac{\#\ of\ Matched\ Relevant\ SlotFillers}{\#\ of\ All\ Matched\ SlotFillers}$$

The procedure proposed in this paper is based on *bunsetsu*s, and an individual *bunsetsu* may contain more than one slot filler. In such cases the procedure is given credit for each slot filler.

Strictly speaking, we don't know how many entities in a matched pattern *might* be slot-fillers when, actually, the pattern does *not* contain any slot-fillers (in the case of over-generating). We approximate the potential number of slot-fillers by assigning 1 if the (falsely) matched pattern does not contain any Named-Entities, or assigning the number of Named-Entities in the (falsely) matched pattern. For example, if we have a pattern "go to dinner" for a management succession scenario and it matches falsely in some part of the test sentences, this match will gain one at the number of All Matched Slot-fillers (the denominator of the precision). On the other hand, if the pattern is "<post> <person> *laugh*" and it falsely matches "President Clinton laughed", this will gain two, the number of the Named Entities in the pattern.

For the sake of comparison, we defined the baseline system with the patterns acquired by the same procedure but only from the direct relationships between a predicate and its arguments (PA in Figure 6 and 7).

We chose the following two scenarios.

- Executive Management Succession: events in which corporate managers left their positions or assumed new ones regardless of whether it was a present (time of the report) or past event.

  Items to extract: Date, person, organization, title.

- Robbery Arrest: events in which robbery suspects were arrested.

  Items to extract: Date, suspect, suspicion.

4.1 Data

	Management Succession
Documents	15
Sentences	79
DATE	43
PERSON	41
ORGANIZATION	22
OLD-ORGANIZATION	2
NEW-POST	30
OLD-POST	39

Table 1: Test Set for Management Succession scenario

	Robbery Arrest
Documents	28
Sentences	182
DATE	26
SUSPICION	34
SUSPECT	50

Table 2: Test Set for Robbery Arrest scenario

For all the experiments, we used the Mainichi-Newspaper-95 corpus for training. As described in the previous section, the system retrieved 300 articles for each scenario as the *relevant document set* from the training data and it further retrieved 300 sentences as the *relevant sentence set* from which all the patterns were extracted.

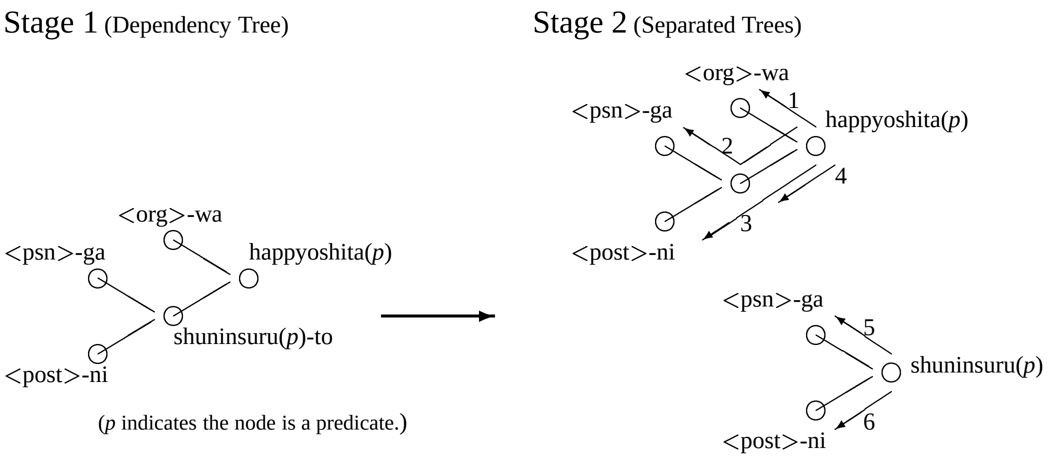
Test data was taken from Mainichi-Newspaper-94 by manually reviewing the data for one month. The statistics of the test data are shown in Table 1 and 2.

4.2 Results

Figure 6 and Figure 7 illustrates the precision-recall curve of this

Japanese sentence :	<organization>-wa	<person>-ga	<post>-ni	shuninsuru-to	happyoshita.
English Translation :	<organization>-TOPIC	<person>-SBJ	<post>-TO	start-COMP	announced.

(<organization> announced that <person> was appointed to <post>.)



Extracted Patterns

1 <organization>-wa → happyosuru

2 <person>-ga → shuninsuru-to → happyosuru

3 <post>-ni → shuninsuru-to → happyosuru

4 shuninsuru-to → happyosuru

5 <person>-ga → shuninsuru

6 <post>-ni → shuninsuru

Figure 4: Pattern Acquisition from "<org>-wa <psn>-ga <pst>-ni shuninsuru-to happyoshita."

experiment for the executive management succession scenario and robbery arrest scenario, respectively. We ranked all the acquired patterns by calculating the sum of the TF/IDF-based score (same as for sentence retrieval in Section 3.3) for each word in the pattern and sorting them on this basis. Then we obtained the precision-recall curve by changing the number of the top-ranked patterns in the list.

Figure 6 shows that TBP is superior to the baseline system both in recall and precision. The highest recall for TBP is 34% while the baseline gets 29% at the same precision level. On the other hand, at the same level of recall, TBP got higher precision (75%) than the baseline (70%).

We can also see from Figure 6 that the curve has a slightly anomalous shape where at lower recall (below 20%) the precision is also low for both TBP and the baseline. This is due to the fact that the pattern lists for both TBP and the baseline contains some non-reliable patterns which get a high score because each word in the patterns gets higher score than others.

Figure 7 shows the result of this experiment on the Robbery Arrest scenario. Although the overall recall is low, TBP achieved higher precision and recall (as high as 30% recall at 40% of precision) than the baseline except at the anomalous point where both TBP and the baseline got a small number of perfect slot-fillers by a highly ranked pattern, namely "*gotoyogi-de* → *taihosuru* (to arrest

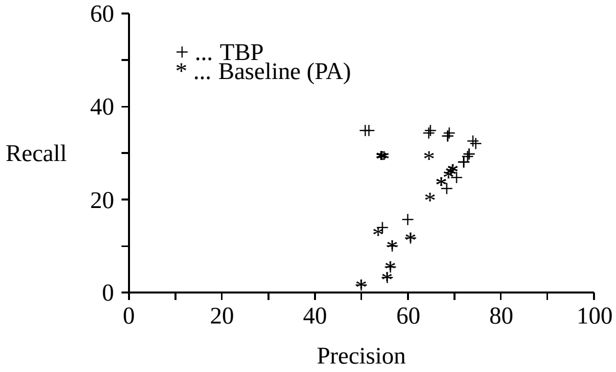


Figure 6: Result on Management Succession Scenario

Scenario	Patterns	
Executive Succession :	<post>-*ni* → *shokakusuru*	(to be promoted to <post>)
	<post>-*ni* → *shuninsuru*	(to assume <post>)
	<post>-*ni* → *shokakusuru* → <jinji>-*o* → *happyosuru*	(to announce an informal decision of promoting somebody to <post>)
Robbery Arrest :	*satsujin-yogi-de* → *taihosuru*	(to arrest in suspicion of murder)
	<date> → *taihosuru*	(to arrest on <date>)
	satsujin-yogi-de → *taihosuru*	(to arrest in suspicion of murder)
	<person>-*yogisha* → *#-o* → *taihosuru*	(to arrest the suspect, <person>, age #)

Figure 5: Acquired Patterns

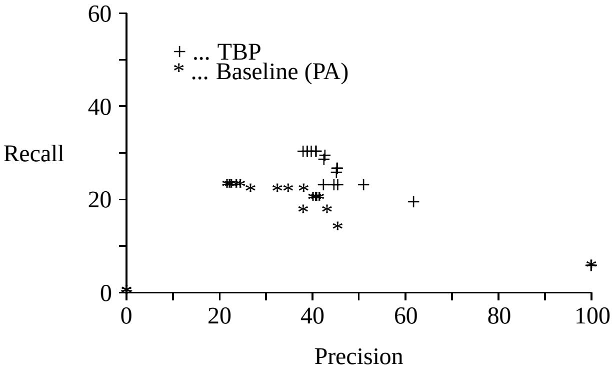


Figure 7: Result on Robbery Arrest Scenario

on suspicion of robbery)” for the baseline and “<person> *yogisha* → <number>-*o* → *taihosuru* (to arrest the suspect, <person>, age <number>)”.

5. DISCUSSION

Low Recall

It is mostly because we have not made a class of types of crimes that the recall on the robbery arrest scenario is low. Once we have a classifier as reliable as Named-Entity tagger, we can make a significant gain in the recall of the system. And in turn, once we have a class name for crimes in the training data (automatically annotated by the classifier) instead of a separate name for each crime, it becomes a good indicator to see if a sentence should be used to acquire patterns. And also, incorporating the classes in patterns can reduce the noisy patterns which do not carry any slot-fillers of the template.

For example on the management succession scenario, all the slot-fillers defined there were able to be tagged by the Named-Entity tagger [5] we used for this experiment, including the *title*. Since we knew all the slot-fillers were in one of the classes, we also knew those patterns whose argument was not classified any of the classes would not likely capture slot-fillers. So we could put more weight on those patterns which contained <person>, <organization>, <post> and <date> to collect the patterns with higher performance, and therefore we could achieve high precision.

Erroneous Case Analysis

We also investigated other scenarios, namely train accident and airplane accident scenario, which we will not report in this paper. However, some of the problems which arose may be worth mentioning since they will arise in other, similar scenarios.

- Results or Effects of the Target Event

  Especially for the airplane accident scenario, most errors were identified as matching the effect or result of the incident. A typical example is “Because of the accident, the airport had been closed for an hour.” In the airplane accident scenario, the performance of the document retrieval and the sentence retrieval is not as good as the other two scenarios, and therefore, the frequency of relevant acquired patterns is rather low because of the noise. Further improvement in retrieval and a more robust approach is necessary.

- Related but Not-Desired Sentences

  If the scenario is specific enough to make it difficult as an IR task, the result of the document retrieval stage may include many documents related to the scenario in a broader sense but not specific enough for IE tasks. In this experiment, this was the case for the airplane accident scenario. The result of document retrieval included documents about other accidents in general, such as traffic accidents. Therefore, the sentence retrieval and pattern acquisition for these scenarios were affected by the results of the document retrievals.

6. FUTURE WORK

Information Extraction

To apply the acquired patterns to an information extraction task, further steps are required besides those mentioned above. Since the patterns are a set of the binary relationships of a predicate and another element, it is necessary to merge the matched elements into a whole event structure.

Necessity for Generalization

We have not yet attempted any (lexical) generalization of pattern candidates. The patterns can be expanded by using a thesaurus and/or introducing a new (lexical) class suitable for a particular domain. For example, the class of expressions of flight number clearly helps the performance on the airplane accident scenario. Especially, the generalized patterns will help improve recall.

Robust Pattern Extraction

As is discussed in the previous section, the performance of our system relies on each component. If the scenario is difficult for the IR task, for example, the whole result is affected. The investigation of a more conservative approach would be necessary.

Translingualism

The presented results show that our procedure of automatic pattern acquisition is promising. The procedure is quite general and addresses problems which are not specific to Japanese. With an appropriate morphological analyzer, a parser that produces a dependency tree and an NE-tagger, our procedure should be applicable to almost any language.

7. ACKNOWLEDGMENTS

This research is supported by the Defense Advanced Research Projects Agency as part of the Translingual Information Detection, Extraction and Summarization (TIDES) program, under Grant N66001-00-1-8917 from the Space and Naval Warfare Systems Center San Diego. This paper does not necessarily reflect the position or the policy of the U.S. Government.

8. REFERENCES

[1] S. Kurohashi and M. Nagao. Kn parser : Japanese dependency/case structure analyzer. In *the Proceedings of the Workshop on Sharable Natural Language Resources*, 1994.

[2] Y. Matsumoto, S. Kurohashi, O. Yamaji, Y. Taeki, and M. Nagano. Japanese morphological analyzing system: Juman. *Kyoto University and Nara Institute of Science and Technology*, 1997.

[3] M. Murata, K. Uchimoto, H. Ozaku, and Q. Ma. Information retrieval based on stochastic models in irex. In *the Proceedings of the IREX Workshop*, 1994.

[4] E. Riloff. Automatically generating extraction patterns from untagged text. In *the Proceedings of Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1996.

[5] S. Sekine, R. Grishman, and H. Shinnou. A decision tree method for finding and classifying names in japanese texts. In *the Proceedings of the Sixth Workshop on Very Large Corpora*, 1998.

[6] R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Unsupervised discovery of scnario-level patterns for information extraction. In *the Proceedings of the Sixth Applied Natural Language Processing Conference*, 2000.